

Database Systems (資料庫系統)

November 26, 2008

Lecture #9

1

Announcement

- TA will talk about midterm exam.
- Assignment #4 will be on the course webpage later today.

2

African Poverb

Every morning, a zebra wakes up,
it knows that it must run faster than
the fastest lion, or it will be eaten;
every morning, a lion wakes up,
it knows that it must run faster than
the slowest zebra, or it will starve to
death;
it does not matter whether you are a
zebra or a lion,
when the sun comes up,
you better run fast.



3

Hash-Based Indexing

Chapter 11

4

Introduction

- Hash-based indexes are best for equality selections. Cannot support range searches.
 - Equality selections are useful for natural join operations.

5

Review: Tree Index

- ISAM vs. B+ trees.
 - What is the main difference between them?
 - Static vs. dynamic structure
 - What is the tradeoff between them?
 - Graceful table growth/shrink vs. concurrency performance

6

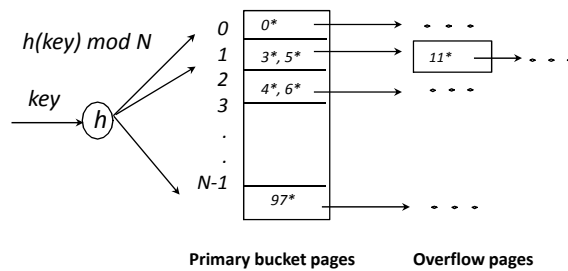
Similar tradeoff for hashing

- Possible to exploit the tradeoff between dynamic structure for better performance?
- Static hashing technique
- Two dynamic hashing techniques
 - Extendible Hashing
 - Linear Hashing

7

Basic Static Hashing

- # primary pages fixed, allocated sequentially, never de-allocated; overflow pages if needed.
- $h(k) \bmod N =$ bucket to which data entry with key k belongs. ($N =$ # of buckets)
 - $N = 2^i$ (or the bucket number is the i -least significant bits)



8

Static Hashing (Contd.)

- Buckets contain data entries.
- Hash function takes search key field of record r and distribute values over range $0 \dots N-1$.
 - $h(\text{key}) = (a * \text{key} + b)$ usually works well.
 - a and b are constants; lots known about how to tune h .
- Cost for insertion/delete/search:
 - 2/2/1 disk page I/Os (no overflow chains).

9

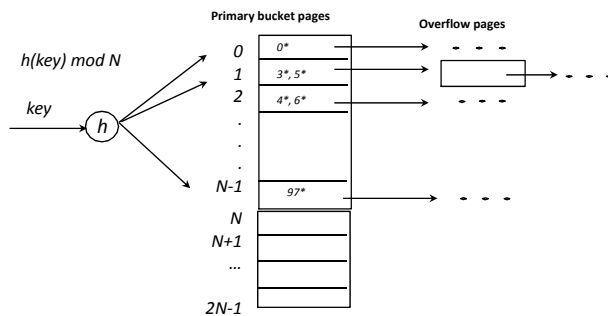
What is the performance problem for static hashing?

- Long overflow chains can develop and degrade performance.
 - Why poor performance? Scan through overflow chains linearly.
- How to fix this problem?
 - Extendible and Linear Hashing: Dynamic techniques to fix this problem.

10

Extendible Hashing

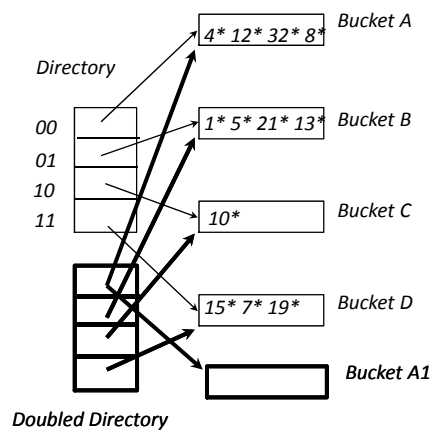
- Simple Solution – remove overflow chain.
- When bucket (primary page) becomes full, ..
 - Re-organize file by adding (doubling) buckets.
- What is the cost concern in doubling buckets?
 - High cost: rehash all entries - reading and writing all pages is expensive!



11

How to minimize the rehashing cost?

- Use another level of indirection
 - Directory of pointers to buckets
- Insert 0 (00)
 - Double the directory (no rehashing)
 - Split just the overflow bucket.
- What is cost here?



12

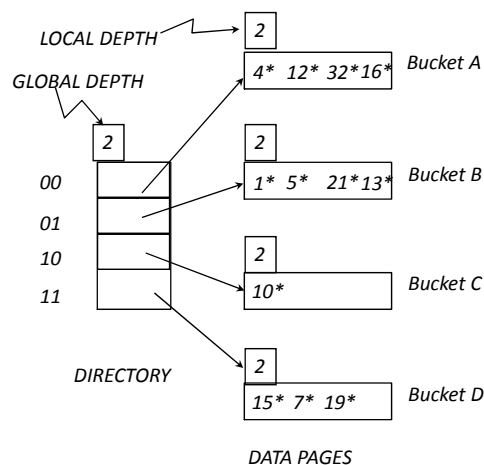
Extendible Hashing

- Directory much smaller than file, so doubling much cheaper. Only one page of data entries is split.
- How to adjust the hash function?
 - Before doubling directory
 - $h(r) \rightarrow 0..N-1$ buckets.
 - After doubling directory?
 - $h(r) \rightarrow 0..2N-1$ (the range is doubled)

13

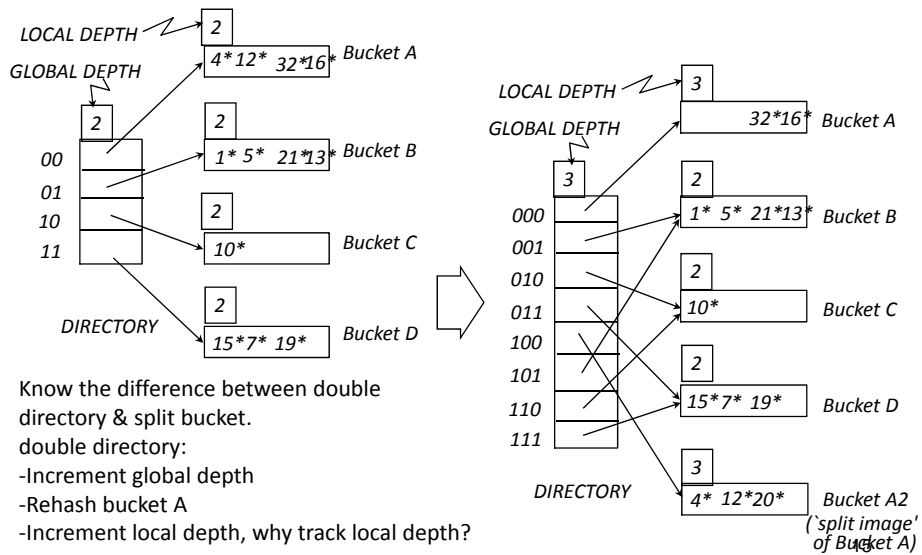
Example

- Directory is array of size 4.
- What info this hash data structure maintains?
 - Need to know current #bits used in the hash function
 - Global Depth
 - May need to know #bits used to hash a specific bucket
 - Local Depth
 - Can global depth < local depth?

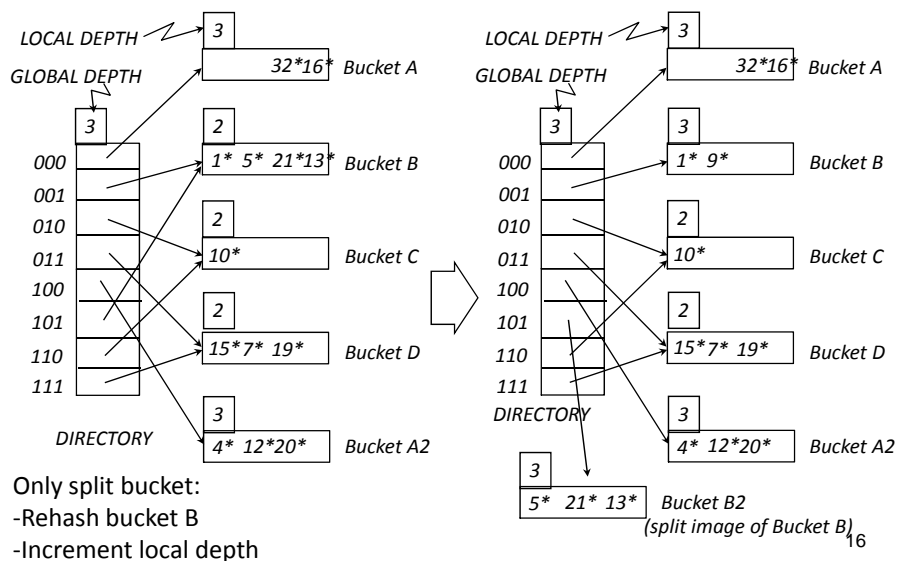


14

Insert 20 (10100): Double Directory



Insert 9 (1001): only Split Bucket



Points to Note

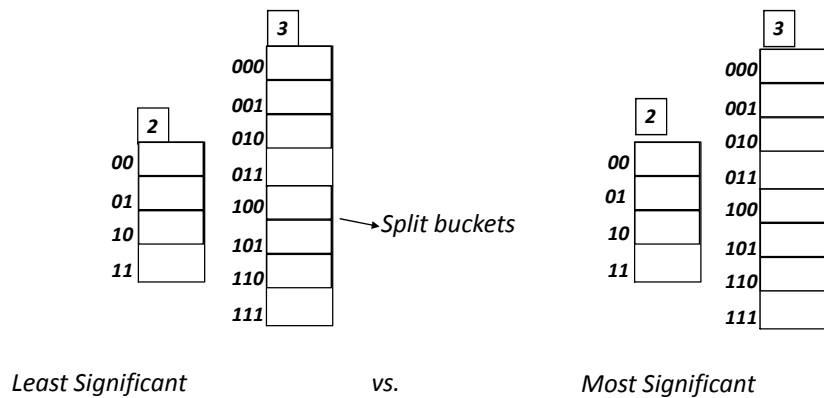
- What is the global depth of directory?
 - Max # of bits needed to tell which bucket an entry belongs to.
- What is the local depth of a bucket?
 - # of bits used to determine if an entry belongs to this bucket.
- When does bucket split cause directory doubling?
 - Before insert, bucket is full & local depth = global depth.
- How to efficiently double the directory?
 - Directory is doubled by copying it over and 'fixing' pointer to split image page.
 - Note that you can do this only by using the least significant bits in the directory.

17

Directory Doubling

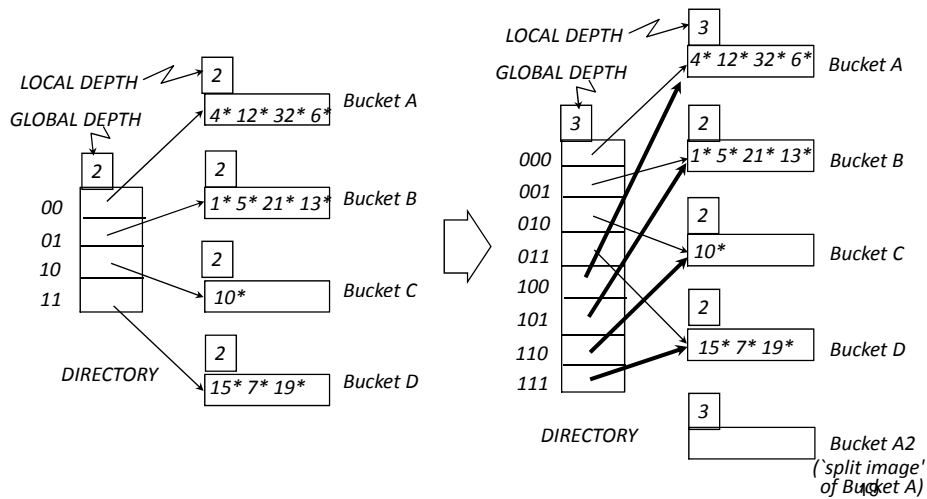
Why use least significant bits in directory?

⇔ Allows for doubling via copying!



18

Directory Copying via Insertion 20 (10100)

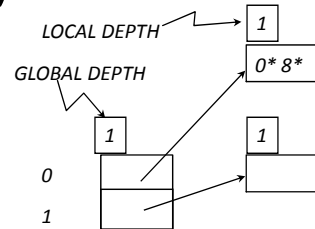


Comments on Extendible Hashing

- If directory fits in memory, equality search answered with one disk access; else two.
- What is the problem with extendible hashing?
 - Consider if the distribution of hash values is skewed (concentrates on a few buckets)
 - Directory can grow large.
- Can you come up with one insertion leading to multiple splits?

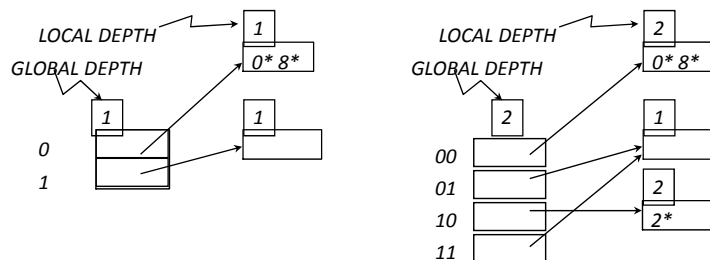
Skewed data distribution (multiple splits)

- Assume each bucket holds two data entries
- Insert 2 (binary 10) – how many times of DIR doubling?
- Insert 16 (binary 10000) – how many times of DIR doubling?
- How to solve this data skew problem?



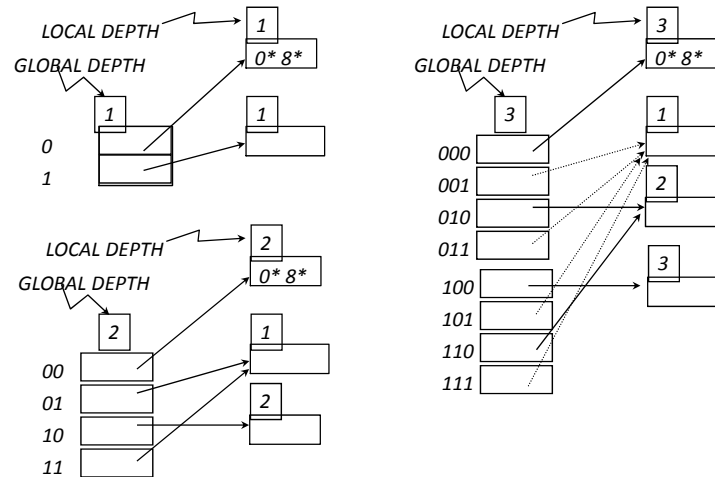
21

Insert 2 (10)



22

Insert 16 (10000)



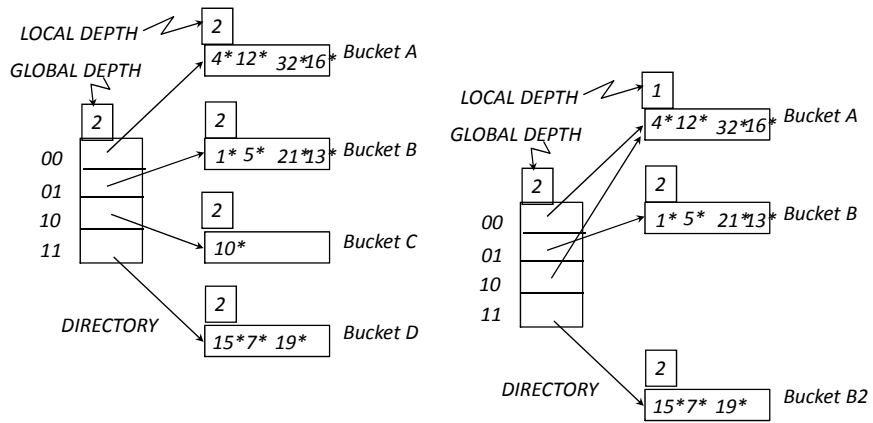
23

Comments on Extendible Hashing

- Delete: If removal of data entry makes bucket empty, can be merged with 'split image'. If each directory element points to same bucket as its split image, can halve directory.

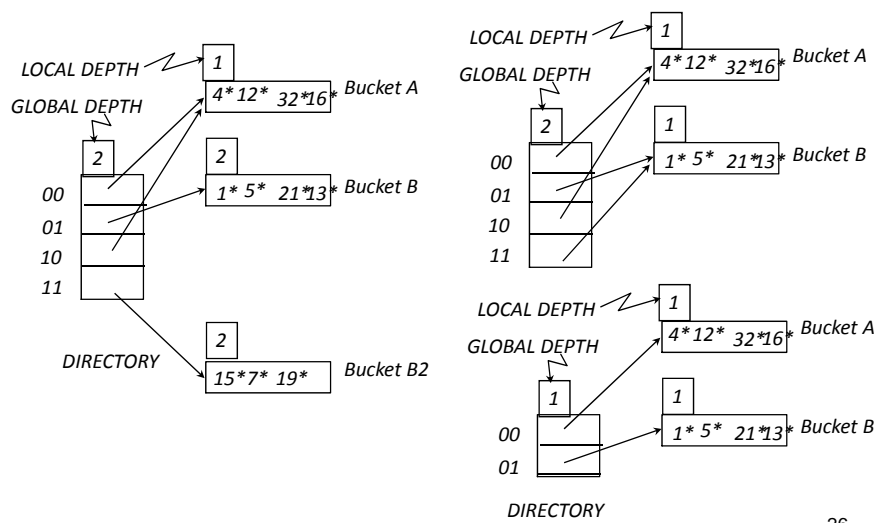
24

Delete 10*



25

Delete $15^*, 7^*, 19^*$



26

Linear Hashing (LH)

- Another dynamic hashing scheme, as an alternative to Extendible Hashing.
- What are problems in static/extendible hashing?
 - Static hashing: long overflow chains
 - Extendible hashing: data skew causing large directory
- Is it possible to come up with a more balanced solution?
 - Shorter overflow chains than static hashing
 - No need for directory in extendible hashing
 - How do you get rid of directory (indirection)?

27

Linear Hashing (LH)

- Basic Idea:
 - Pages are split when overflows occur – but not necessarily the page with the overflow.
 - Splitting occurs in turn, in a round robin fashion.
 - one by one from the first bucket to the last bucket.
- Use a family of hash functions h_0, h_1, h_2, \dots
 - Each function's range is twice that of its predecessor.
 - When all the pages at one level (the current hash function) have been split, a new level is applied.
 - Splitting occurs gradually
 - Primary pages are allocated in order & consecutively.

28

Linear Hashing Verbal Algorithm

- Initial Stage.
 - The initial level distributes entries into N_0 buckets.
 - Call the hash function to perform this h_0 .

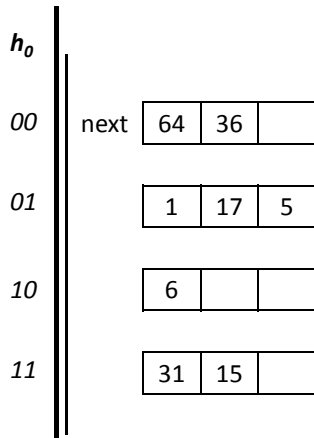
29

Linear Hashing Verbal Algorithm

- Splitting buckets.
 - If a bucket overflows its primary page is chained to an overflow page (same as in static hashing).
 - Also when a bucket overflows, some bucket is split.
 - The first bucket to be split is the first bucket in the file (*not* necessarily the bucket that overflows).
 - The next bucket to be split is the second bucket in the file ... and so on until the N th. has been split.
 - When buckets are split their entries (including those in overflow pages) are distributed using h_1 .
 - To access split buckets the next level hash function (h_1) is applied.
 - h_1 maps entries to $2N_0$ (or N_1) buckets.

30

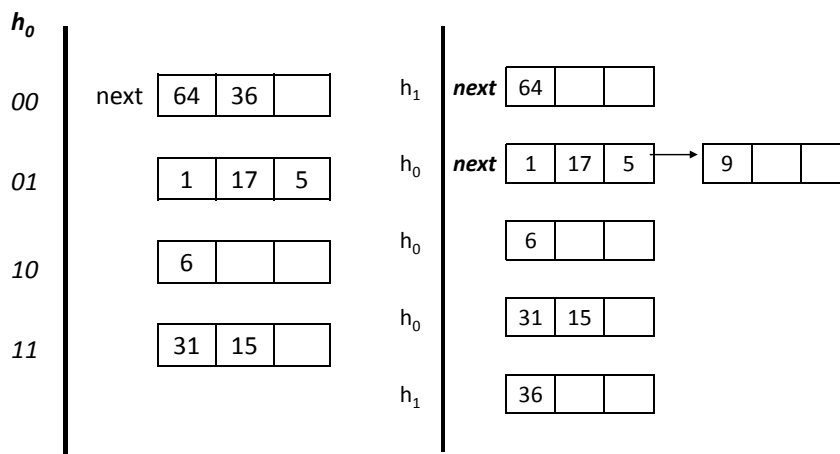
Linear Hashing Example



- Initially, the index level equal to 0 and N_0 equals 4 (three entries fit on a page).
- h_0 range = 4 buckets
- Note that **next** indicates which bucket is to split next. (Round Robin)
- Now consider what happens when 9 (1001) is inserted (which will not fit in the second bucket).

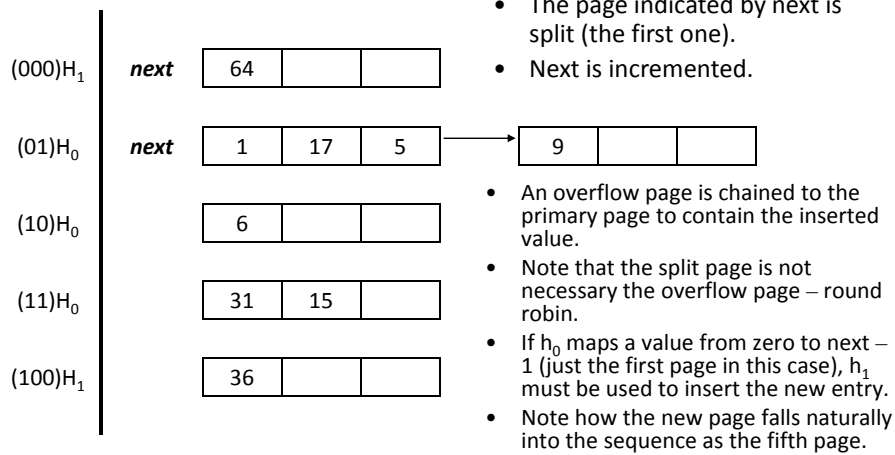
31

Insert 9 (1001)



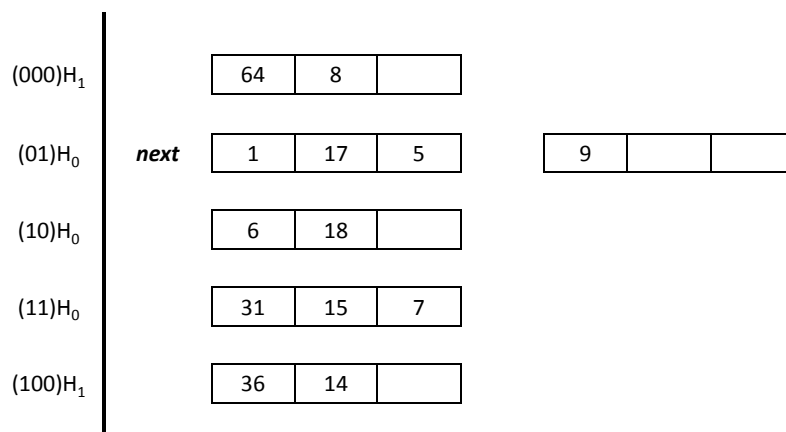
32

Linear Hashing Example 2



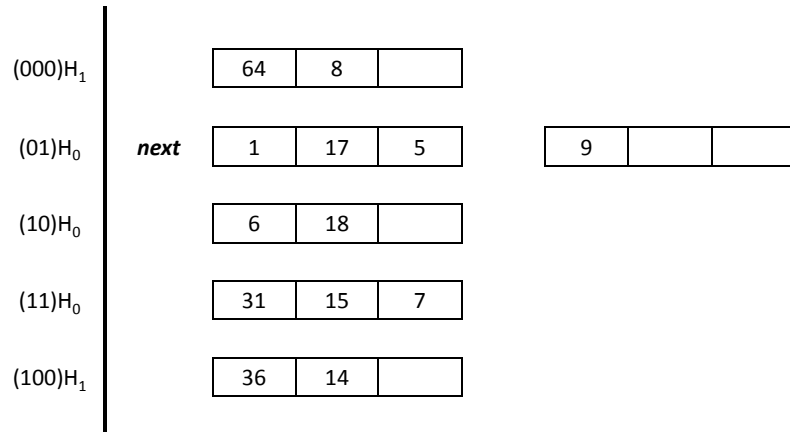
33

Insert 8 (1000), 7(111), 18(10010), 14(1100)



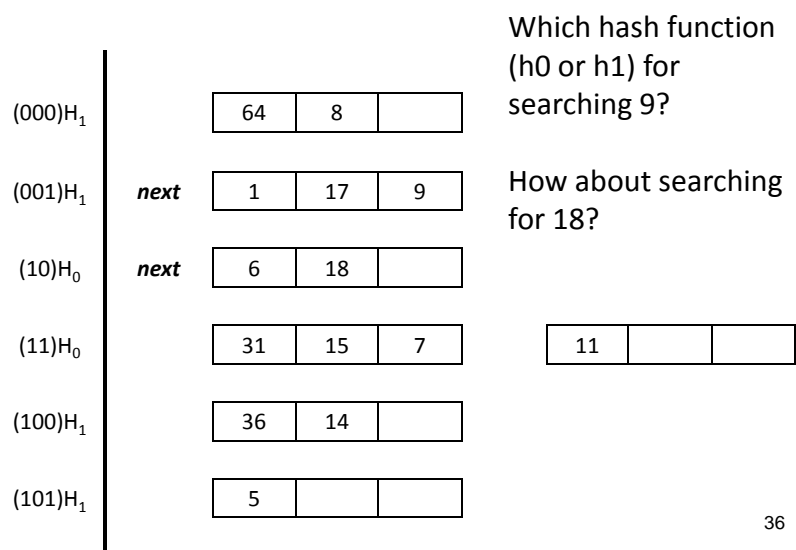
34

Before Insert 11 (1011)



35

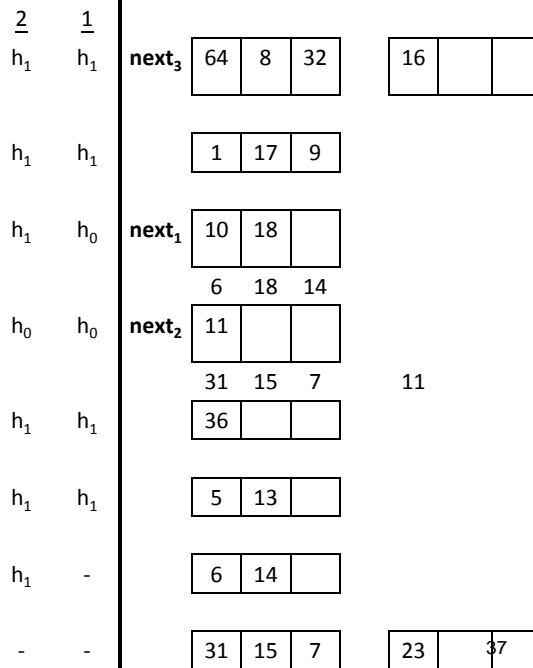
After Insert 11 (1011)



36

Linear Hashing

- Insert 32, 16₂
- Insert 10, 13, 23₃
- After the 2nd. split the base level is 1 ($N_1 = 8$), use h_1 .
- Subsequent splits will use h_2 for inserts between the first bucket and *next-1*.



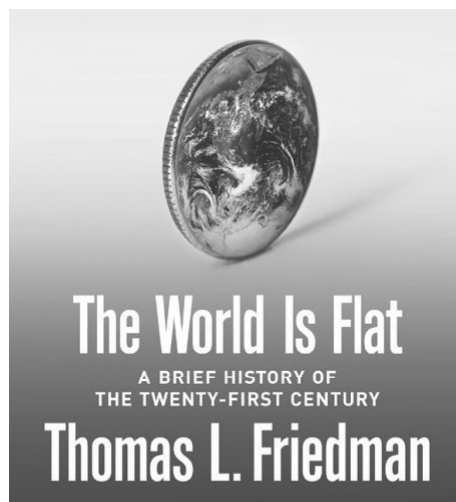
Notes for Linear Hashing

- Why linear hashing does not need a directory but extendible hashing needs one?
 - Consider finding i-th bucket
 - New buckets are created in order (sequentially, i.e., round robin) in linear hashing
- Level progression:
 - Once all N_i buckets of the current level (i) are split the hash function h_i is replaced by h_{i+1} .
 - The splitting process starts again at the first bucket and h_{i+2} is applied to find entries in split buckets.

LH Described as a Variant of EH

- Can you describe linear hashing using extendible hashing (EH)?
 - Begin with an EH index where directory has N elements.
 - Use overflow pages, split buckets round-robin.
 - First split is at bucket 0. (Imagine directory being doubled at this point.) But elements $\langle 1, N+1 \rangle, \langle 2, N+2 \rangle, \dots$ are the same. So, need only create directory element N , which differs from 0, now.
 - When bucket 1 splits, create directory element $N+1$, etc.
- So, directory can double gradually. Also, primary bucket pages are created in order. If they are *allocated* in sequence too (so that finding i 'th is easy), we actually don't need a directory!
Voila, LH.

39



40

The New World is Flat

- With computer and network, the playing field is being leveled across the world.
- There is no end to what can be done by whom
 - Before: manufacturing products (Made in Taiwan, Made in China)
 - Now: anything including service industry
- This is called outsourcing
- The rule of Capitalist economy:
 - Work goes to the place where it can be done cheapest and most efficient

41

Outsourcing Examples

- Customer services moving to call centers in India
 - Credit cards, banks, insurance, Microsoft, ...
- Radiologists in India and Australia
- E-tutoring
- Jetblue & homesourcing
- Tax preparation outsourcing
- MacDonald Flatfries and burgers
- Kenichi Ohmae (大前研一)

42

Outsourcing and ME in Taiwan?

- So anyone on this planet may be replaced by anyone on this planet for cheaper and more efficiency
 - Yes, for any jobs – high/low tech, manufacturing/service
 - Scaring? Consider massive amount of smart & cheap labors in China and India
- Oops ... this is happening NOW here in Taiwan!

- So one can compete or collaborate globally with anyone on this planet -> opportunity.

43